Paper 023-30

Connecting Simply with SAS/CONNECT®

Michael G. Sadof, MGS Associates, Inc.

ABSTRACT

Since it is common to have data residing across several platforms, it is important to utilize the strengths of each computer system. SAS/CONNECT® Software enables you to view and update your data on multiple platforms from any one connected computer. This paper provides some quick and easy code to get your SAS/CONNECT applications up and running smoothly.

With only a few simple statements you can manipulate your UNIX or mainframe based data from your PC environment without batch processing. Similarly you may submit and run UNIX SAS programs from an MVS batch job. The data may actually reside on either platform.

With business or research data residing on various platforms the SAS system utilizes the client server model to access the data wherever it resides. SAS/CONNECT Software is one tool for us to view, edit, manipulate or copy data from one platform to another. The SAS/CONNECT software must be installed on each platform. It is beyond the scope of this paper to describe the installation procedures but with one exception (the RLINK script) they are transparent to the user. When your installation is complete you will have several script files available to assist in the connection process. The scripts may require some minor adjustments but for the most part function very well. Once installed the user has the option of processing, saving, or copying data locally or remotely. Thus you can take advantage of the strengths of the various platforms but have a convenient and familiar interface.

DEFINITIONS

First, let us start with some definitions. The Local Computer or Local Host Computer is the computer to which you are initially signed on. For instance if you utilize PC SAS and you open up a SAS session then your PC SAS session is the Local Host. If you use your PC to log into a UNIX SAS session through terminal emulator software then your Local Host is the UNIX session. When you use either of these local SAS sessions (interactive or batch) to sign on to another SAS session on a different platform that is the Remote Host or Remote Server. When we send data from the Local Computer to the Remote Host we are uploading. When we are copying data from a Remote Computer or Internet server to our local machine we are downloading. The PROC DOWNLOAD procedure will move stored data from the Remote Host to your Local Session. The PROC UPLOAD procedure will move locally stored data to the Remote Host. Remember that the Local Session may not be your PC it may in fact be the UNIX SAS session if you have established the SAS session via Telnet or the mainframe batch job that you have submitted. The Server referred to in this paper is the machine that is running the SAS Software. As we will see, it is possible to nest remote sessions to accomplish your goals and in that case you may have to identify the remote server. Figure 1 depicts a typical SAS Connect network design and shows the relationship between the Local Host and the Remote Computer.

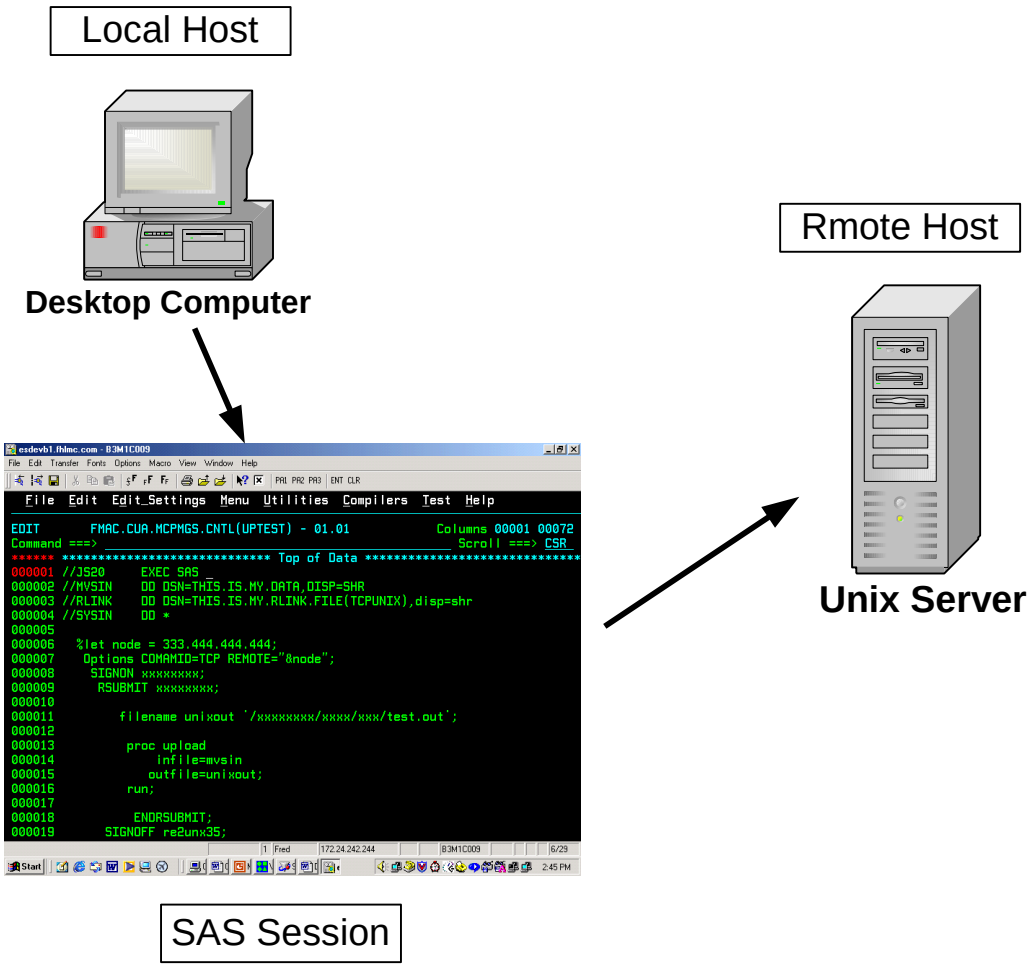

FIGURE 1.

FIVE EASY STEPS

There are essential five steps in every SAS/CONNECT session.

1. Establish the communication access method.
2. Identify the RLINK script file
3. Sign on
4. Remote submit your program/ End Remote Submit
5. Sign Off

Step 1

The communications access method is the protocol by which the two computers will communicate. The most common access method seems to be TCP/IP and will work for SAS/CONNECT just as it does for Internet communications. The communications access method or COMAMID is specified in the OPTIONS statement. To use the TCP/IP communications access method you simply specify that in the OPTIONS statement as follows:

```
OPTIONS COMAMID=TCP REMOTE=xxx;
```

In the OPTIONS statement the **'xxx'** parameter identifies the IP address of the computer to which you want to communicate. Most times the network administrator has set up an alias or DNS name for the computer you are looking for. You must know the exact name or IP address to set up the communication link. Sometimes sample programs are available on you system from which you could learn the IP address or alias. The value of the REMOTE= options must be a valid SAS name so sometimes it is convenient to code it as follows:

```
%LET node=123.2233.444.333;
OPTIONS COMAMID=TCP REMOTE="&node";
```

Step 2

A script file that is provided by SAS accomplishes the actual linking up of the two computers. This script file must be identified to the SAS program with the FILENAME statement. These script files are generally found in the following directory on your PC with the 'scr' extension:

```
c:\program files\sas institute\sas\v8\connect\saslink\
```

An example of the RLINK file identification is as follows where the '!sasroot' has been identified in the configuration file when SAS was installed:

```
filename rlink "!sasroot\connect\saslink\tcpunix.scr";
```

The script for linking to a UNIX session is 'tcpunix.scr' and the script for linking up to a mainframe TSO session is 'tcptso.scr'. Check your "!sasroot\connect\saslink" directory for further options. The script file is a text file containing comments and commands to log on to the remote computer. Sometimes this file must be tweaked to allow it perform properly. To adjust this file you must first set 'echo on;' and 'trace on;'. These commands within the file will produce traces and echoes to the SAS log so you can determine where the problem lies. In my experience the most problems occur with the timing so you may want to bump up some of the 'wait' times. Another area of concern is the command that actually starts SAS on the remote machine. You will note that the 'tcpunix.scr' script file (for UNIX) a command similar to the following will actually start SAS:

```
type 'sas -dmr -comamid tcp -device grlink -noterminal -nosyntaxcheck' LF;
```

You may have to adjust this command to point to your actual SAS exe file on UNIX like:

```
/opt/bin/local/sas612/sas.exe
```

Again, your network administrator or SAS administrator will have a better idea of how to utilize the scripts. Your mainframe computer will have a similar library of scripts to logon to UNIX or NT. Specifying the RLINK file on MVS can be done in the JCL or the FILENAME statement.

Step 3

The actual signon is accomplished with the signon statement and is the statement in your SAS program that initiates the script in the RLINK file. It is handy to code it with the node name as follows:

```
OPTIONS REMOTE=sunsas COMAMID=TCP;

FILENAME RLINK "!SASROOT\connect\saslink\tcpunix.scr";

SIGNON sunsas;
```

Step 4

The execution of programs on the remote machine is accomplished within an RSUBMIT block which is a pair of commands like a {DO;END;} block.

```
OPTIONS REMOTE=sunsas COMAMID=TCP ;
FILENAME RLINK "!sasroot\connect\saslink\tcpunix.scr";
SIGNON sunsas;

RSUBMIT;
    ****;
    * SAS statements;
    ***;
ENDRSUBMIT:
```

All statements within the RSUBMIT block will be executed on the remote machine. So if you fire up your PC SAS session and submit a program which will log you on to a UNIX SAS session you can use the UNIX machine to compute or access data using the PC interface. Within the RSUBMIT block all of your SAS commands will be executed on the remote machine. You may define libraries with the LIBNAME statement or files with the FILENAME statement and process data as if you were on the UNIX machine directly.

Step 5

To end to remote session the signoff procedure is as simple as coding the SIGNOFF command as follows:

```
SIGNOFF;
```

REMOTE COMPUTE SERVICES

After establishing a connection to the remote host all commands within the RSUBMIT; ENDRSUBMIT; block are processed on the remote computer and comprise the remote compute services functionality of SAS/CONNECT. You may enclose several procedures or data steps within the RSUBMIT block. You may then return back to the host by issuing statements outside of the RSUBMIT block and then return to the remote system by enclosing commands in another RSUBMIT block.

FILE TRANSFER SERVICES

SAS/CONNECT has two procedures to copy files between the remote host and the Local session. As mentioned earlier the download procedure will bring data from the remote host to your local session. The upload procedure will move data from the local session to the remote host. For instance, if we have established a local PC SAS session and then connected over to UNIX we can use the download procedure to copy data from UNIX to our local PC session. PROC DOWNLOAD has several options to copy a variety of catalogs, libraries, SAS files or external non-SAS files. As with PROC SORT the PROC DOWNLOAD has a syntax which generally requires DATA= and OUT= clauses.

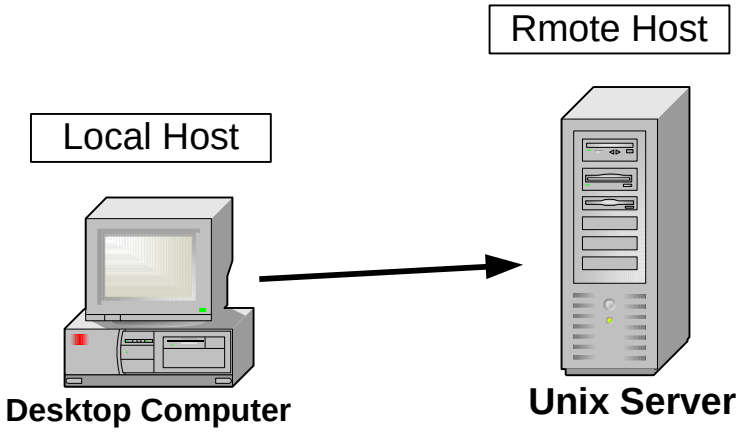

FIGURE 2.

To copy a SAS data file from remote to local you would issues the following statements:

```
PROC DOWNLOAD
   DATA=ONE
   OUT=TWO;
```

If you omit the OUT= clause the data will be copied to 'WORK.ONE' on the local machine. It always seems clearer to define both libraries explicitly rather than to let SAS use the default. PROC DOWNLOAD has several pairs of options with are used with various files:

DATA=/OUT=	- For individual SAS data sets.
INLIB=/OUTLIB=	- For SAS libraries (may be multiple data sets)
INCAT=/OUTCAT=	- For SAS catalogs
INFILE=/OUTFILE=	- For text or binary files

Each form of the DOWNLOAD procedure permits various options. For instance the DATA=/OUT= form permits the 'where clause' so you could issue the following commands to select only certain records;

```
PROC DOWNLOAD DATA=ONE
     OUT=TWO;
WHERE X=1;
```

To move complete or partial libraries from the host to the local machine we might submit the following commands. Notice that the local library must be defined outside of the RSUBMIT block. In this example only two members (cars, trucks) of the library are copied.

```
LIBNAME local_library 'c:\';

RSUBMIT;
   LIBNAME remote_library '/mysasdir/';
    PROC DOWNLOAD INLIB=remote_library
                  OUTLIB=local_library;
    SELECT cars trucks;
ENDRSUBMIT;
```

External files, whether they be text files, program files, or binary files can be copied from the remote machine with the PROC DOWNLOAD procedure using the INFILE=/OUTFILE= specification. Usually the files are defined by a FILENAME statement but may be defined directly:

```
FILENAME remote1 '/unix/sample/file.txt';
FILENAME remote2  '/unix/sample/program.exe';

PROC DOWNLOAD INFILE=remote1
              OUTFILE='c:/myfiles/sample.txt';

PROC DOWNLOAD INFILE=remote2
              OUTFILE='c:/myfiles/program.exe' / BINARY;
```

To send data files, program files, or SAS data files from the local machine to the remote host use the PROC UPLOAD procedure. The syntax is similar to the download procedure.

REMOTE LIBRARY SERVICES

Once you have established a SAS/CONNECT session and have entered into an RSUBMIT block you can define a remote library with the LIBNAME statement. Once defined you may access that library as if it were connected to your local machine. It will appear in the explorer window with a special icon indicating it is a remote library. When using Remote Library Services you define the remote library to your local session with the SLIBREF= and SERVIER= clauses. The syntax for defining a remote library is as follows:

```
OPTIONS REMOTE=sunsas COMAMID=TCP;
FILENAME RLINK "C:\sas\connect\saslink\tcpunixf.scr";

signon sunsas;

RSUBMIT sunsas;
   LIBNAME  myremote "/your/unix/library/";
endrsubmit;

* Now on the local Session *;
LIBNAME newremote
    SLIBREF=myremote
    SERVER=sunsas;

PROC CONTENTS DATA=newremote._all_;
```

An explanation of the above code is as follows:
The OPTIONS statement will assign the TCP communications access method and name the remote session to be 'sunsas'. This must be a valid alias for the particular server's IP address. The FILENAME statement will select the logon script file to be used. The SIGNON statement will begin execution of the RLINK script file. After execution of the SIGNON statement you will see the logon messages from the remote computer and you will in fact be logged on to the remote SAS session as well as your local session. The RSUBMIT command will indicate that all of the statements until the ENDRSUBMIT command will be executed on the remote host. The LIBNAME statement will allocate a remote library to the remote session. It will not be available to your local session until you ENDRSUBMIT and go back to the local session and then execute the last LIBNAME statement. This final LIBNAME statement will assign a library name of 'newremote' to you local session but it will point to the 'myremote' library on the remote host. The SLIBREF= clause must point to the remote name and the remote server. At this point you may use PROC COPY or any other SAS procedure to copy or manipulate the data on your local

machine. You must remember, however, that the data is residing on the remote machine and network traffic might delay a procedure. Using this method you will be able to process remote data on your local machine as opposed to submitting commands to process the data on the remote machine. Your local SAS session will retrieve data from the remote computer as necessary to process SAS statements. This processing is done outside of the RSUBMIT block as shown in the above example. . One added benefit of SAS/CONNECT is that is can read and write both version 6.12 and version 8 SAS files. You can connect from a Version 8 system directly to a version 6.12 system and the translation will be transparent to the user.

SAS CONNECT SPAWNER

Setting up SAS connect services on your Windows box (NT Server) can be easy and productive, First of course you must have a SAS Connect License for your server. The Server license for SAS Connect comes with the 'spawner.exe' program which can be installed as a background service on Windows NT/200/XP Professional.

INSTALL SPAWNER AS A SERVICE:

The system administrator or someone with admistrative privileges must run the spawner program as follows:

```
spawner -service sas_spawner -install -servuser ".\uuu" -servpass ppp -nosecurity
```

(where ***uuu*** is the administrator userid and ***ppp*** is the password)

```
net start "SAS JOB SPAWNER"
```

SERVICES FILE:

Place and entry like the following in the \etc\services file to utilize port 5555 for a TCP/IP Listening port: (I can be placed anywhere in the file and it is fine to place it after the last entry)

```
sas_spawner      5555/tcp      # SAS Spawner
```

The program spawner.exe will listen on that port for any SAS commands and process any commands received. On windows NT/2000/XP the services file is located at:

```
C:\WINDOWS\SYSTEM32\DRIVERS\ETC\SERVICES
```

LOGON FROM SAS

Logon to the spawner from a SAS session with a program as described previously:

```
filename rlink "!sasroot\connect\saslink\tcpwin.scr";

%let node=localhost 5555;     * where localhost can be any valid DNS *;
                              * name on your network                 *;

options comamid=tcp remote=node ;

signon node ;
```

Please note that when identifying the remote server you must indicate the DNS name or IP address along with the port number where the listener is installed. In the example 5555 is the port number identified in the services file entry as described in **SERVICES FILE** above

CONCLUSION

SAS/CONNECT Software can assist in distributing the processing load on your computer network. You may use these services to connect to less heavily loaded computers on your network and obtain faster results. It enables you to view remote data easily from within your local SAS session or copy files reliably. SAS/CONNECT gives you the flexibility to control where you data is stored and processed. It enables you to work with a more familiar interface whether it is in the PC, UNIX or Mainframe batch environment. It is particularly handy when SAS data is spread across multiple platforms and you would like to work with it in one program. The Remote Library services, Remote Compute services, and the File Transfer services bring it all together in one nice package known as SAS/CONNECT.

APPENDIX

SAMPLE PROGRAM

```
*************************************;
* Define Rlink File                 *;
*************************************;

filename rlink "!sasroot\connect\saslink\tcpunx.scr" ;

*************************************;
* Define any Local Libraries        *;
* These are libraries on your machine *;
*************************************;

libname mgs v6 'c:\My Documments\My SAS Files\';

*************************************;
* Set up options on Local Machine    *;
* Name remote session                *;
*************************************;

%let node1=myserver.mynode.com;

*************************************;
* Use port number if signing on to   *;
* SAS Spawner                        *;
*************************************;

* %let node1=myserver.mynode.com 4444 ; * where 4444 is port number;

options comamid=TCP remote="&node" ;

*************************************;
* Signon to remote session           *;
* name session if you wish or are using *;
* more than one session concurrently  *;
* ---------------------------------- ;
* This will execute your logon script  *;
* as defined by the RLINK file above   *;
* Modify RLINK file if necessary       *;
* Signon name must match remote name   *;
*************************************;
signon Node1;

*************************************;
* Submit commands to remote machine   *;
* via the RSUBMIT command             *;
*************************************;
```

```
rsubmit Node1;

    ******************************************;
    * Sign on to second remote through first    *;
    *   SIGN-ON TO UNIX (SUN) FROM NT SERVER    *;
    ******************************************;

    options comamid=TCP remote=Node2;

    filename rlink "C:\sas\connect\saslink\tcpunixf.scr" ;

    ************************************;
    * Assign Library on first Node        *;
    ************************************;

    libname first_node_lib "/users/mike/sas/directory";

    signon Node2;

       rsubmit Node2;

          ************************************************;
          * allocate libraries or files on remote machine    *;
          ************************************************;

          libname  node2_lib   "/users/my_rem_lib/my_sas_files/";
          filename node2_file  "/users/my_rem_lib/data_file.dat";
          run;

       endrsubmit; ** Node2 **;

      ************************************;
      * Back on Node 1                     *;
      * allocate remote libraries on Node1 *;
      * with remote library services       *;
      ************************************;

        libname rem_node2
                slibref=node2_lib
                server=Node2;

        libname saved "/users/my_current_lib/";

          ** Manipulate data on Unix for example;

          data rem_node2.one;
          x=1;output;run;

          ** Download to local if necessary **;

          proc download data=node2_lib.testd
                               out=saved.dat1;

  endrsubmit; ** Node1 **;

****************************;
* Back on Local             *;
* allocate remote library   *;
* on local host pointing    *;
* to Node2 through Node1    *;
****************************;

libname lib_one   slibref=first_node_lib   server=Node1;
```

```
proc print data=lib_one.test;

signoff ; ** Node2 ** ;
signoff ; ** Node1 ** ;
```

REFERENCES

SAS Institute Inc., SAS/CONNECT® Software: Usage and Reference, Version 6, Second Edition, Cary, NC: SAS Institute Inc., 1994

TRADEMARKS

SAS® and SAS/CONNECT® are registered trademarks of the SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

Please feel free to contact the author at:

Michael G. Sadof
MGS Associates, Inc.
Bedford, NH
mgs@mgsnet.net